Flex-Sphere: An FPGA Configurable Sort-Free Sphere Detector For Multi-user MIMO Wireless Systems

Kiarash Amiri, (Rice University, Houston, TX, USA; kiaa@rice.edu);
Chris Dick, (*Advanced Systems Technology Group (ASTG)*, Xilinx, San Jose, CA, USA; chris.dick@xilinx.com);
Raghu Rao, (*Advanced Systems Technology Group (ASTG)*, Xilinx, San Jose, CA, USA; raghu.rao@xilinx.com);
Joseph R. Cavallaro, (Rice University, Houston, TX, USA; cavallar@rice.edu)

***Abstract*—Spatial division multiplexing (SDM) in MIMO technology significantly increases the spectral efficiency, and hence capacity, of a wireless communication system: it is a core component of the next generation wireless systems, e.g. WiMAX, 3GPP LTE and other OFDM-based communication schemes. Moreover, spatial division multiple access (SDMA) is one of the widely used techniques for sharing the wireless medium between different mobile devices. Sphere detection is a prominent method of simplifying the detection complexity in both SDM and SDMA systems while maintaining BER performance comparable with the optimum maximum-likelihood (ML) detection. There are several approaches for realizing sphere detectors, and the algorithmic landscape is rich with methods that enable the designer to make various tradeoffs between performance, e.g. throughput of the wireless channel, BER, and implementation complexity, e.g. silicon area for an ASIC implementation or FPGA resource envelope for an FPGA implementation. This paper describes the FPGA realization of a *configurable* and flexible sort-free sphere detector, *Flex-Sphere*, that supports $4,16,64$-QAM modulations as well as a combination of $2,3$ and 4 antenna/user configuration for uplink transmission. The detector provides a data rate of up to 849.9 Mbps. The algorithmic optimizations employed to produce an FPGA friendly realization are discussed.**

I. Introduction

Multiple-input multiple-output (MIMO) communication systems and spatial division multiplexing (SDM) have recently drawn significant attention as a means to achieve tremendous gains in system capacity and link reliability. Moreover, spatial division multiple access (SDMA) has recently received attention for its promise to increase the sum data rate of different users in wireless networks, and creating a virtual MIMO between multiple users and a base station.

The optimal hard decision detection, in terms of BER performance, for all MIMO wireless systems is the maximum likelihood (ML) detector. However, direct implementation of ML grows exponentially with the number of antennas and the modulation scheme, making its ASIC or FPGA implementation infeasible for all but low-density modulation schemes using a small number of antennas. Sphere detection (SD) solves the ML detection problem in a computationally efficient manner [1], [2], [3], [4], [5].

This paper reports on the FPGA implementation of a configurable and flexible sphere detector called Flex-Sphere. Flex-Sphere supports three commonly used modulation schemes, $4,16,64$-QAM, as well as a combination of $2,3$ and 4 user/antenna configuration. The detector provides a data rate of up to 849.9 Mbps. The bread-first search employed in our realization presents a large opportunity to exploit the parallelism of the FPGA in order to achieve high data rates. Algorithmic modifications to address potential sequential bottlenecks in the traditional bread-first search-based SD are highlighted in the paper.

The optimizations and tradeoffs employed to minimize the FPGA resource utilization in the metric computation units are also described along with performance curves that quantify the BER performance of these methods that are directed at minimizing the cost of the FPGA implementation. We also describe the model-based design flow that was used to produce the FPGA implementation.

The rest of the paper is organized as follows: Section II presents the general system model. The proposed FPGA friendly architecture for the sort-free MIMO detector is presented in section III. Section IV introduces the model-based design of the Flex-Sphere using Xilinx System Generator. The simulation results for floating point and FPGA fixed point of the system for different parameters are given in section V. Finally, the paper concludes with section VI.

II. System Model

We assume a *virtual* MIMO system with n tranmitters each with $L_r, r = 1,...,n$ antennas such that $M_T = \Sigma_{r=1}^{n} L_r$, and a receiver, e.g. a basestation, with $M_R \geq M_T$ receive antennas. All the transmitters use the same channel to communicate simultaneously with the receiver. The input-output model is captured by

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\tilde{\mathbf{s}} + \tilde{\mathbf{n}} \tag{1}$$

where $\tilde{\mathbf{H}}$ is the complex-valued $M_R \times M_T$ channel matrix, $\tilde{\mathbf{s}} = [\tilde{s}_1, \tilde{s}_2, ..., \tilde{s}_{M_T}]^T$ is the M_T-dimensional transmitted vector from the n transmitters, where each $\tilde{s}_j, j = 1,...,M_T$, is chosen from a complex-valued constellation Ω_j of the order $w_j = |\Omega_j|$, $\tilde{\mathbf{n}}$ is the circularly symmetric complex additive white Gaussian noise vector of size M_R and $\tilde{\mathbf{y}} = [\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{M_R}]^T$ is the M_R-element received vector. Note that we do not restrict all the parallel M_T streams to use the same modulation order; rather,

each stream, which corresponds to one of the antennas of one of the users, may be using either the $4, 16$ or 64-QAM modulation.

The preceding MIMO equation can be decomposed into real-valued numbers as follows [6]:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{2}$$

corresponding to

$$\begin{pmatrix} \Re(\tilde{\mathbf{y}}) \\ \Im(\tilde{\mathbf{y}}) \end{pmatrix} = \begin{pmatrix} \Re(\tilde{\mathbf{H}}) & -\Im(\tilde{\mathbf{H}}) \\ \Im(\tilde{\mathbf{H}}) & \Re(\tilde{\mathbf{H}}) \end{pmatrix} \begin{pmatrix} \Re(\tilde{\mathbf{s}}) \\ \Im(\tilde{\mathbf{s}}) \end{pmatrix} + \begin{pmatrix} \Re(\tilde{\mathbf{n}}) \\ \Im(\tilde{\mathbf{n}}) \end{pmatrix} \tag{3}$$

with $M = 2.M_T$ and $N = 2.M_R$ presenting the dimensions of the new model.

We call the ordering in (2), the conventional ordering. Using the conventional ordering, all the computations can be performed in only real values. Note that after real-valued decomposition, each $s_i, i = 1, ..., M$, in $\mathbf{s}$ is chosen from a set of real numbers, Ω_i', with $w_i' = \sqrt{w_i'}$ elements. For instance, for a 64-QAM modulation, each s_i can take any of the values in the set $\Omega' = \{\pm 7, \pm 5, \pm 3, \pm 1\}$.

The general optimum detector for such a system is the maximum-likelihood (ML) detector which minimizes $\| \mathbf{y} - \mathbf{H}\mathbf{s} \|^2$ over all the possible combinations of the $\mathbf{s}$ vector. Notice that for high order modulations and large number of antennas, this detection scheme incurs an exhaustive exponentially growing search among all the candidates, and is not practically feasible in a MIMO receiver. However, it is shown that using the QR decomposition of the channel matrix, the distance norm can be simplified [7] as follows:

$$\begin{aligned} D(\mathbf{s}) &= \| \mathbf{y} - \mathbf{H}\mathbf{s} \|^2 \\ &= \| \mathbf{Q}^{\mathbf{H}}\mathbf{y} - \mathbf{R}\mathbf{s} \|^2 = \sum_{i=M}^{1} |y_i{}' - \sum_{j=i}^{M} R_{i,j} s_j|^2 \end{aligned} \tag{4}$$

where $\mathbf{H} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q}\mathbf{Q}^{\mathbf{H}} = \mathbf{I}$ and $\mathbf{y}' = \mathbf{Q}^{\mathbf{H}}\mathbf{y}$. Note that the transition in (4) is possible through the fact that $\mathbf{R}$ is an upper triangular matrix.

The norm in (4) can be computed in M iterations starting with $i = M$. When $i = M$, i.e. the first iteration, the initial partial norm is set to zero, $T_{M+1}(\mathbf{s}^{(M+1)}) = 0$. Using the notation of [3], at each iteration the Partial Euclidean Distances (PEDs) at the next levels are given by

$$T_i(\mathbf{s}^{(i)}) = T_{i+1}(\mathbf{s}^{(i+1)}) + |e_i(\mathbf{s}^{(i)})|^2 \tag{5}$$

with $\mathbf{s}^{(i)} = [s_i, s_{i+1}, ..., s_M]^T$, and $i = M, M-1, ..., 1$, where

$$|e_i(\mathbf{s}^{(i)})|^2 = |y_i{}' - R_{i,i}s_i - \sum_{j=i+1}^{M} R_{i,j}s_j|^2 \tag{6}$$

$$= |b_{i+1}(\mathbf{s}^{(i+1)} - R_{i,i}s_i|^2. \tag{7}$$

One can envision this iterative algorithm as a tree traversal with each level of the tree corresponding to one i value, and each node having w_i' children.

The tree traversal can be performed in a breadth-first manner. At each level, only the best K nodes, i.e. the K nodes with the smallest T_i, are chosen for expansion. This type of detector is generally known as the K-best detector. Note that such a detector requires sorting a list of size $K \times w'$ to find the best K candidates. For instance, for a 16-QAM system with $K = 10$, this requires sorting a list of size $K \times w' = 10 \times 4 = 40$ at most of the tree levels. This introduces a long delay for the next processing block in the detector unless a highly parallel sorter is used. Highly parallel sorters, on the other hand, consist of a large number of compare-select blocks, and result in dramatic area increase.

III. Flex-Sphere SDM/SDMA Detector

In order to simplify the sorting step, which significantly reduces the delay of the detector, we propose a novel MIMO detector. This detector is based on a sort-free strategy, and utilizes a new modified real-valued decomposition ordering (M-RVD) scheme.

A. Tree Traversal for Sort-Free Detection

In order to address the sorting challenge, we propose using a sort-free detector. With this technique, the long sorting operation is effectively simplified to a minimum-finding operation. The detailed steps of this algorithm are described below:

Input: $\mathbf{R}$, $\mathbf{y}'$
$T_{M+1}(\mathbf{s}^{(M+1)}) = 0$
$\mathcal{L} \leftarrow \emptyset$
$\mathcal{L}' \leftarrow \emptyset$
$i \leftarrow M$
\\ Full expansion of the first level:
- Compute T_i with (5),
- $\mathcal{L} \leftarrow \{(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j | j = 1, ..., w_i'\}$
- $i \leftarrow i - 1$
\\ Full expansion of the second level:
- **for** each $(\mathbf{s}^{(i+1)}, T_{i+1}(\mathbf{s}^{(i+1)})) \in \mathcal{L}$, **repeat**
  - compute $(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j$ children pairs, $j = 1, ..., w_i'$
  - $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j | j = 1, ..., w_i'\}$
- **end**
-$\mathcal{L} \leftarrow \mathcal{L}'$
-$\mathcal{L}' \leftarrow \emptyset$
\\ Minimum-based expansion of the next levels:
- **for** $i = M - 2$ down to $i = 1$, **repeat**
  - **for** each $(\mathbf{s}^{(i+1)}, T_{i+1}(\mathbf{s}^{(i+1)})) \in \mathcal{L}$, **repeat**
    - compute $(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j$ children pairs, $j = 1, ..., w_i'$
    - $(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_{min} \leftarrow \underset{\{(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j | j = 1, ..., w_i'\}}{\operatorname{argmin}} T_i(\mathbf{s}^{(i)})$
    - $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_{min}\}$
  - **end**
  - $\mathcal{L} \leftarrow \mathcal{L}'$
  - $\mathcal{L}' \leftarrow \emptyset$
  - $i \leftarrow i - 1$
- **end**

- $(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_{detected} \leftarrow \underset{\mathcal{L}}{\operatorname{argmin}} \; T_i(\mathbf{s}^{(i)})$

An example of this algorithm is illustrated in Figure 1 for a virtual 4×4, 64-QAM system. Note that as described above, the first two levels are fully expanded to guarantee high performance; whereas for the following levels, only the best candidate in the children list of a parent node is expanded. In other words, after passing the first two levels, w_{M_T} nodes are expanded, and for each of those w_{M_T} nodes, the best children node among its w'_M children nodes are selected as the survived node. Therefore, the new node list would contain w_{M_T} nodes in the third level. These w_{M_T} nodes are expanded in a similar way to the forth level, and this procedure continues until the very last level, where the minimum-distance node is taken as the detected node.

Moreover, from the Schnorr-Euchner (SE) ordering [8], we know that finding

$$(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_{min} \leftarrow \underset{\{(\mathbf{s}^{(i)}, T_i(\mathbf{s}^{(i)}))_j | j=1,\ldots,w'_i\}}{\operatorname{argmin}} T_i(\mathbf{s}^{(i)})$$

basically corresponds to finding the real-valued constellation point closest to $\frac{1}{R_{ii}} b_{i+1}(\mathbf{s}^{(i+1)})$; see Eq. (7). Thus, the long sorting of K-best is avoided.

B. Modified Real-Valued Decomposition (M-RVD) Ordering

For the sort free detector described in the preceding section, we propose using a novel real-valued decomposition (M-RVD) ordering which improves the BER performance compared to the ordering given in Eq. (2). The new decomposition is summarized as:

$$\hat{\mathbf{y}} = \hat{\mathbf{H}}\hat{\mathbf{s}} + \hat{\mathbf{n}} \tag{8}$$

or,

$$\begin{pmatrix} \Re(\tilde{y}_1) \\ \Im(\tilde{y}_1) \\ \Re(\tilde{y}_2) \\ \Im(\tilde{y}_2) \\ . \\ . \\ . \\ \Re(\tilde{y}_{M_R}) \\ \Im(\tilde{y}_{M_R}) \end{pmatrix} = \hat{\mathbf{H}} \begin{pmatrix} \Re(\tilde{s}_1) \\ \Im(\tilde{s}_1) \\ \Re(\tilde{s}_2) \\ \Im(\tilde{s}_2) \\ . \\ . \\ . \\ \Re(\tilde{s}_{M_T}) \\ \Im(\tilde{s}_{M_T}) \end{pmatrix} + \begin{pmatrix} \Re(\tilde{n}_1) \\ \Im(\tilde{n}_1) \\ \Re(\tilde{n}_2) \\ \Im(\tilde{n}_2) \\ . \\ . \\ . \\ \Re(\tilde{n}_{M_R}) \\ \Im(\tilde{n}_{M_R}) \end{pmatrix} \tag{9}$$

where $\hat{\mathbf{H}}$ is the permuted channel matrix of Eq. (3) whose columns are reordered to match the other vectors of the new decomposition ordering in Eq. (8). It is worth noting that there is no extra computational cost associated with this novel ordering.

Note that with the modified real-valued decomposition (M-RVD) ordering, the first two levels correspond to the in-phase and quadrature parts of the same complex symbol; whereas in the conventional real-valued decomposition scenario, the first two levels of the tree correspond to the quadrature parts of two different complex symbols.

IV. SYSTEM GENERATOR FPGA DESIGN

In this section, the main features of the FPGA implementation are presented. We use Xilinx System Generator [9] to implement the proposed architecture. In order to support all the different number of antenna/user and modulation orders, the detector is designed for the maximal case, i.e. 4×4, 64-QAM case, and configurability elements are introduced in the design to support different configurations.

A. PED Computations

Computing the norms in (7) is performed in the PED blocks. Depending on the level of the tree, three different PED blocks are used: The PED in the first real-valued level, PED_1, corresponds to the root node in the tree, $i = M = 2M_T = 8$. The second level consists of $\sqrt{64} = 8$ parallel PED_2 blocks, which compute 8 PEDs for each of the 8 PEDs generated by PED_1; thus, generating 64 PEDs for the $i = 7$ level. Followed by this level, there 8 parallel general PED computation blocks, PED_g, which compute the closest-node PED for all 8 outputs of each of the PED_2s. The next levels will also use PED_g. At the end, the Min_Finder unit detects the signal by finding the minimum of the 64 distances of the appropriate level. The block diagram of this design is shown in Figure 2.

B. Configurable Design

In order to ensure the configurability of the Flex-Sphere, it needs to to support different number of antenna/users as well as different modulation orders for different users.

1) Number of User/Antenna: The number of user/antenna, M_T, determines the number of detection levels, and it is set through M_T input to the detector, which in turn, would configure the Min_Finder appropriately. Therefore, the minimum finder can operate on the outputs of the corresponding level, and generate the minimum result. In other words, the multiplexers in each input of the Min_Finder block, choose which one of the four streams of data should be fed into the Min_Finder. Therefore, the inputs to the Min_Finder would be coming from the $i = 5, 3$ or 1, if M_T is $2, 3$ or 4; respectively, see Figure 2.

Note that the M_T input can change on-the-fly; thus, the design can shift from one mode to another mode based on the number of streams it is attempting to detect at anytime. Moreover, as will be shown later, the configurability of the minimum finder guarantees that less latency is required for detecting smaller number of streams.

2) Modulation Order: In order to support different modulation orders per data stream, the Flex-Sphere uses another input control signal $q^{(i)}$ to determine the maximum real value of the modulation order of the i-th level. Thus, $q^{(i)} \in \{1, 3, 7\}$. Moreover, since the modulation order of each level is changing, a simple comparison-thresholding can not be used to find the closest candidate for Schnorr-Euchner [8] ordering. Therefore, the following conversion is used to find the closest SE candidate:

$$\tilde{s} = g(2\lfloor \frac{b+1}{2} \rfloor - 1) \tag{10}$$

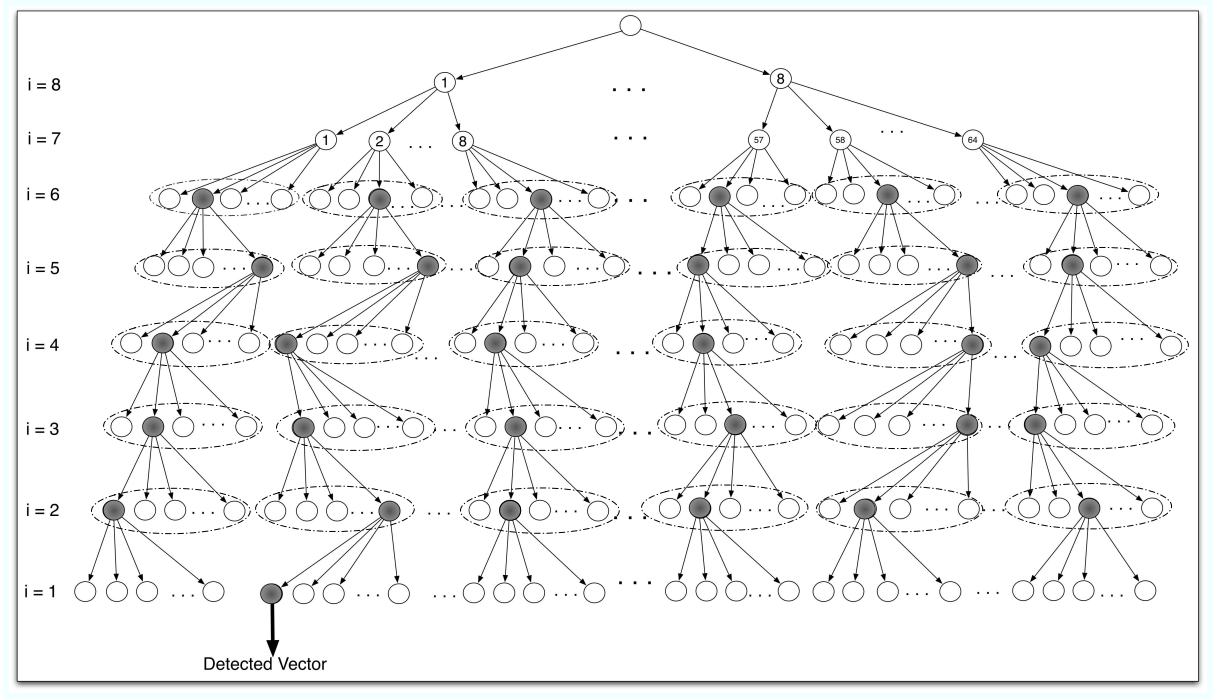

Fig. 1. Sort-free algorithm for a 64-QAM, 4×4 system. The topmost two levels are fully expanded. The nodes marked with black are the minimum in their own set, where each set is denoted by dashed line. Note that because of the real-valued decomposition, each node has only $\sqrt{64} = 8$ children. Also, the number of tree levels are $M = 2 \times M_T = 8$.

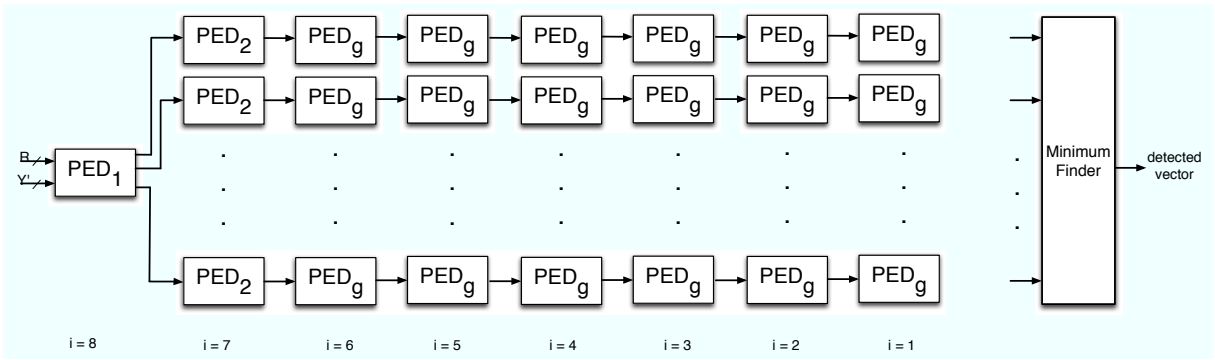

Fig. 2. The block diagram of the Flex-Sphere. Note that there are 8 parallel PEDs at each level. The inputs to the Min_Finer is fed from the appropriate PED block, as described in section IV-B1.

where $[.]$ represents rounding to the nearest integer, $b = (1/R_{ii}) \cdot b_{i+1}$ of Eq. (7), and $g(.)$ is

$$g(x) = \begin{cases} -q^{(i)} & x \leq -q^{(i)} \\ x & -q^{(i)} \leq x \leq q^{(i)} \\ q^{(i)} & x \geq q^{(i)} \end{cases} \tag{11}$$

All of these functions can be readily implemented using the available building blocks of the Xilinx System Generator, see Figure 3. Note that the multiplications/divisions are simple one-bit shifts.

For the first two levels, which corresponds to the in-phase and quadrature components of the last antenna, the PED of the out-of-range candidates are simply overwritten with the maximum value; thus, they will be automatically discarded during the minimum-finding procedure.

C. Modified Real Valued Decomposition (M-RVD)

Using the real-valued decomposition, the two extra adders that are required per each complex multiplication, can be avoided; thus, avoiding the unnecessary FPGA slices on the addition operations. Moreover, while using the complex-valued operations require the SE ordering of [3], which would be a demanding task given the configurable nature of the detector; with the real-valued decomposition, the SE ordering can be implemented more efficiently and simply for the proposed configurable architecture as described earlier. Also,

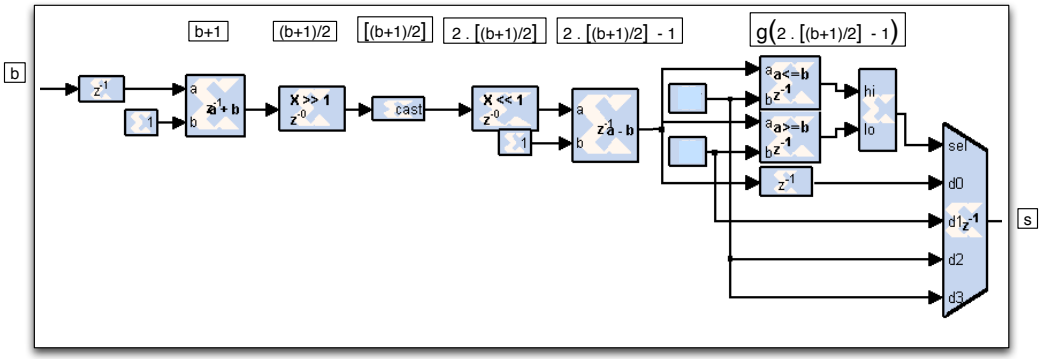

Fig. 3. System Generator block diagram for Eq. (10) in the PED_g to support different modulation orders.

Note that even though some of the multiplications can be replaced with shift-adds in an area-optimized ASIC design; for an FPGA implementation, the appropriate deign choice is to use the available embedded multipliers, commonly known as XtremeDSP and DSP48E in Virtex-4 and Virtex-5 devices.

It is noteworthy that if the conventional real-valued decomposition of (3) was employed; then, the results for a 2×2 system would have been ready only after going through all the in-phase tree levels and the first two quadrature levels; whereas, with the modified real-valued decomposition (M-RVD), since every antennas is isolated from other antennas in two consecutive levels of the tree, there is no need to go through the latency of the unnecessary levels. Thus, using the M-RVD technique, offers a latency reduction compared to the conventional real-valued decomposition.

D. Timing Analysis

Each of the PED_g blocks are responsible for expanding 8 nodes; thus, the folding factor of the design is $F = 8$. In order to ensure a high maximum clock frequency, several pipelining levels are introduced inside each of the PED computation blocks. The latency of the PED_1, PED_2 and PED_g blocks are $7, 17$ and 22, respectively. Note that the larger latency of the PED_g blocks is due to the more multiplications required to compute the PEDs of the later levels. The Min_Finder block has a latency of 8.

Note that as mentioned earlier, different values of M_T require different number of tree levels; thus, incurs different latencies. The latencies of the three different configurations of M_T are presented in Table I. Note that in computing the latencies, an initial 8 cycles are required to fill up the pipeline path.

TABLE I
LATENCY FOR DIFFERENT VALUES OF M_T.

M_T	Latency
$M_T = 2$	$8 + \text{PED}_1 + \text{PED}_2 + 2 \cdot \text{PED}_g + \text{Min\_Finder} = 84$
$M_T = 3$	$8 + \text{PED}_1 + \text{PED}_2 + 4 \cdot \text{PED}_g + \text{Min\_Finder} = 128$
$M_T = 4$	$8 + \text{PED}_1 + \text{PED}_2 + 6 \cdot \text{PED}_g + \text{Min\_Finder} = 172$

E. Implementation Results

Figure 4 presents the System Generator implementation of the Flex-Sphere detector. Table II presents the implementation results of the Flex-sphere on a Xilinx Virtex-5 FPGA, xc5vsx95t-3ff1136 [9] for 16-bits precision. The maximum achievable clock frequency is 283.3 MHz. Since the design folding factor is set to $F = 8$, the maximum achievable data rate, i.e. $M_T = 4$ and $w_i = 64$, is

$$D = \frac{M_T \cdot \log w}{F} \cdot f_{max} = 849.9 \text{ [Mbps]}. \tag{12}$$

Table III summarizes the data rates for all of the different scenarios.

TABLE II
FPGA RESOURCE UTILIZATION SUMMARY OF THE PROPOSED FLEX-SPHERE FOR THE XILINX VIRTEX-5, XC5VSX95T-3FF1136, DEVICE.

Number of Slices	12,443/14,720 (84 %)
Number of Slice Registers	37,759/58,880 (64 %)
Number of Look-Up Tables	27,803/58,880 (47 %)
Number of DSP48E	273/640 (42 %)
Max. Freq.	283.3 MHz

TABLE III
DATA RATE FOR DIFFERENT CONFIGURATIONS.

	4-QAM	16-QAM	64-QAM
$M_T = 2$	141.6 Mbps	283.3 Mbps	424.9 Mbps
$M_T = 3$	212.4 Mbps	424.9 Mbps	637.4 Mbps
$M_T = 4$	283.3 Mbps	566.6 Mbps	849.9 Mbps

V. SIMULATION RESULTS

In this section, we present the simulation results for the Flex-Sphere. Throughout this section, we assume that the channel ordering of [10] is performed prior to the detection in the pre-processing section. Also, we make the assumption that all the streams are using the same modulation scheme. We assume complex-valued channel matrices, with the real and imaginary parts of each element drawn from the normal

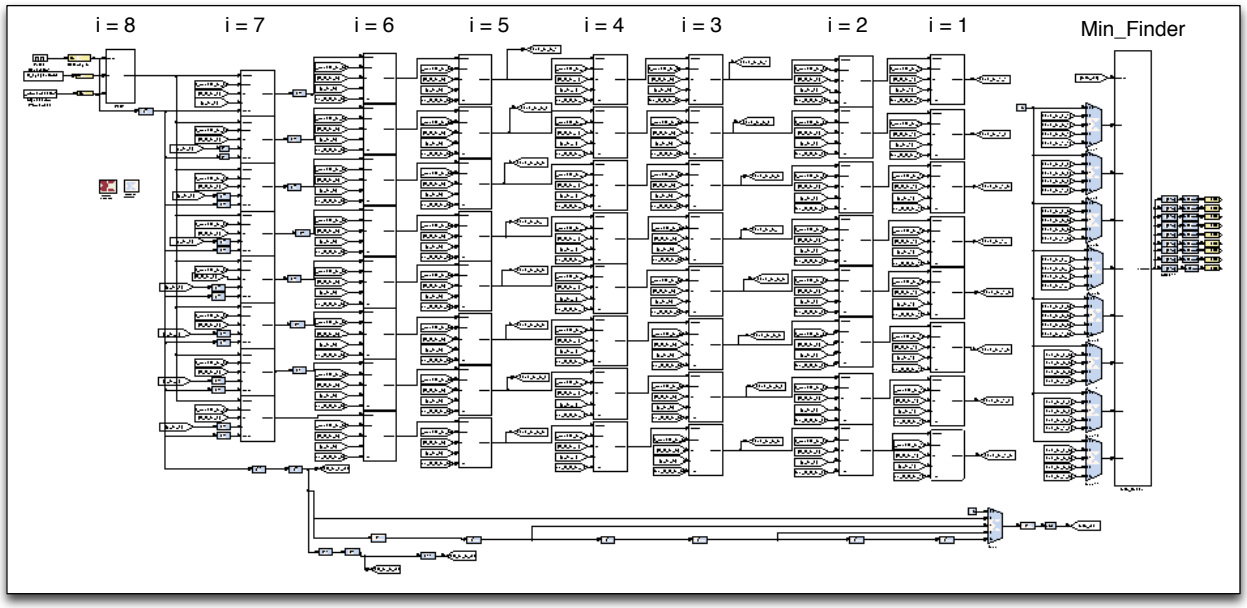

Fig. 4. Xilinx System Generator implementation of the Flex-Sphere detector.

distribution. In order to ensure that all the antennas in the receiver have similar average received SNR, and none of the users messages are suppressed with other messages, a power control scheme is employed. Figure 5 shows the simulation results for the 64-QAM, 4×4 configuration.

VI. CONCLUSION

In this paper, we reported on a configurable and flexible multi-user MIMO detector, which can support different number of antennas and modulation orders required by a wide variety of different standards. We presented the FPGA implementation results, and the simulation results suggest that the performance is considerably close to the optimum ML detector.

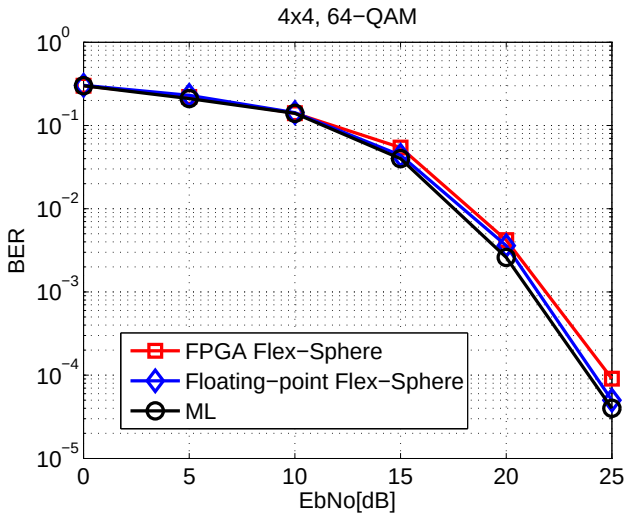

Fig. 5. BER plots comparing the 4×4, 64-QAM of the maximum likelihood (ML) with the MATLAB floating point simulations and the System Generator implementation.

REFERENCES

[1] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Computat.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[2] E. Viterbo and J. Boutros, "A universal lattice decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[3] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE JSSC*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[4] K. Amiri and J. R. Cavallaro, "FPGA implementation of dynamic threshold sphere detection for MIMO systems," *40th Asilomar Conf on Signals, Systems and Computers*, Nov 2006.

[5] M. Jiang, S. X. Ng, and L. Hanzo, "Hybrid iterative multiuser detection for channel coded space division multiple access OFDM systems," *IEEE Transactions on Vehicular Technology*, vol. 55, pp. 115–127, Jan 2006.

[6] Z Guo and P. Nilsson, "A 53.3 Mb/s 4×4 16-QAM MIMO decoder in 0.35μm CMOS," *IEEE Int. Symp. Circuits Syst.*, vol. 5, pp. 4947–4950, May 2005.

[7] M. O. Damen, H. E. Gamal and G. Caire, "On maximum likelihood detection and the search for the closest lattice point," *IEEE Trans. on Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[8] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, no. 2, pp. 181–191, Sep. 1994.

[9] "Xilinx : https://www.xilinx.com/."

[10] L. G. Barbero and J. S. Thompson, "A fixed-complexity MIMO detector based on the complex sphere decoder," *Signal Processing Advances in Wireless Communications, 2006. SPAWC '06. IEEE 7th Workshop on*, Jul. 2006.